

ECE 4331 Final Project

Part 1: (You may use Mathcad for all of your simulations/calculations)

- Design a digital, low-pass filter based on a 4th-order Butterworth low-pass filter with cutoff frequency $\omega_c = 200\pi$ rad/sec (i.e., $f_c = 100$ Hz).
- Employ the bilinear transformation to the analog filter to obtain the digital filter's transfer function, $H(z)$. Assume a sampling period of $T_s = 10^{-3}$ sec. Is frequency pre-warping necessary?
- Determine and plot the digital filter's poles and zeros.
- Determine the filter's recursion formula for the output $y[k]$.
- Employ Mathcad and the Laplace transform to solve for the zero-state response of the analog filter, $y_{zs}(t)$ assuming the input $f(t) = \cos(100\pi t)u(t) + \cos(300\pi t)u(t)$.
- Employ the recursion formula from Part d to solve (numerically) for the digital filter's zero-state response, $y_{zs}[k]$, assuming the sampled input signal $f(T_s k)$.
- Provide a single plot comparing the shifted input component $\cos(100\pi(t + t_d))u(t)$ to the digital filter response (from Part f) and analog filter response (from Part e). Here, t_d represents the analog filter delay at 50Hz. Choose the time-axis so as to display about three periods of the signals.

Part 2:

- Implement your digital filter on an Arduino microcontroller. You accomplish this by programming the filter's recursion formula. The input to one of the ADC (analog to digital convertor) pins of your microcontroller should be the sum (use an op-amp adder) of two in-phase sinusoids with frequencies $f_1 = 50$ Hz and $f_2 = 150$ Hz. The sinusoids are to be generated by a signal generator*. The microcontroller reads the ADC, processes this value using the recursion formula and outputs through an R2R DAC (digital to analog convertor) every $T = 1$ ms. Here are useful links to help you build the R2R convertor:

<https://www.arduino.cc/en/Main/Software>
<https://www.instructables.com/id/R2R-Digital-Analog-Converter-DAC/>
https://www.youtube.com/watch?v=_tABS7AX8D8&feature=youtu.be

Employ 10K and 20K 1% tolerance resistors to build your R2R circuit. The output of the DAC convertor and the 50 Hz sinusoid are to be monitored on an oscilloscope. The results from this step must be included in your report.

Adding a 0.47uF capacitor between the output of the R2R convertor and ground leads to a smoother the digital filter signal. Try it.

Capture the scope displays depicting the digital filter output for the following input signals (with and without the smoothing capacitor):

- a. $f(t) = \cos(100\pi t) + \cos(300\pi t)$
 - b. $f(t) = \cos(100\pi t) + \cos(200\pi t)$
 - c. $f(t) = \cos(100\pi t) + \cos(1200\pi t)$
 - d. $f(t) = \cos(100\pi t) + \cos(2000\pi t)$
 - e. $f(t)$ is a 1Vpp square wave with 20Hz frequency.
2. Build an analog 4th order Butterworth low-pass filter with cut-off frequency of 100 Hz (you may employ Matlab and “afd” software to determine the component values). Use the scope to display the output of the analog filter alongside that of the digital filter (employ a 0.47uF capacitor at the output of the digital filter to smooth it):
- a. $f(t) = \cos(100\pi t) + \cos(300\pi t)$
 - b. $f(t) = \cos(100\pi t) + \cos(200\pi t)$
 - c. $f(t) = \cos(100\pi t) + \cos(1200\pi t)$
 - d. $f(t) = \cos(100\pi t) + \cos(2000\pi t)$
 - e. $f(t)$ is a 1Vpp square wave with 20Hz frequency.

Note: the DAC of the Arduino accepts only positive input between 0 and 5 volts. Therefore, you need to bias the input sinusoids by adding 1Volt dc to each. This bias will restrain the input $f(t)$ to be between 0 and 4 volts. Since the filter is low-pass, the dc bias will propagate with no attenuation. In turn, the output values of the recession formula (digital filter) will have no negative values and will be compatible with the DAC; The R2R convertor can only handle positive values. So, when displayed on the scope, the output of the R2R and the input signal will both have the same amount of bias. You can eliminate this bias by setting the scope channels to AC.

*The sinusoids generated by the signal generator must be synced. Also, the signal generator sinusoid is a sine wave. To get a cosine wave you will have to add a 90 degrees phase shift.

[Watch Video](#)